Scientific Computing

Installation of Python using Conda and first steps in Python

Peter Regner, Johannes Schmidt

Institute for Sustainable Economic Development, BOKU, Wien

2020-03-26

Latest version available on Github, this PDF is version 74366ae.

1. Organizational matters
2. Conda
3. Jupyter
4. Homework assignment

Organizational matters

Organizational matters

- Please participate in writing today's lecture notes:
  https://yourpart.eu/p/lecture-scientific-computing02-notes

Organizational matters

- Please participate in writing today's lecture notes: https://yourpart.eu/p/lecture-scientific-computing02-notes
- Let's also try to write a glossary of new terms: https://yourpart.eu/p/lecture-scientific-computing-glossary
- Testing and Grading
  - Do your homework in your group! We'll check repositories. (Relevant for grading)
  - Presentation at least once in the online lecture - either a homework exercise or a lecture exercise.
  - Review tests at the start of each class (starting in coming week). Relevant for you, not for grading.

Conda

Download...

Please download Anaconda from
https://www.anaconda.com/distribution/

Package manager and Environments

- A package (library or application) is a collection of code that helps you accomplish tasks (a bit similar to the `setup.exe` in Windows).
- There are e.g. packages for machine learning, for plotting data, for working with tabular data, or for working with matrix-data. More on that later!
- A package manager is a convenient way to install software: it resolves all dependencies (=other packages needed by the package you want to install), downloads all packages and installs them.
- Conda also allows to create environments: within an environment, you are free to install a different Python version, different packages and package versions etc. This helps in having a clean separation between projects. (But also means that you may have installed Python many times on your computer)

Package Manager of our choice

- We are going to use conda as package manager.
- Anaconda is a system that packages a lot of software together with conda and runs on Windows and Linux. It also has a graphical user interface. In class, we use the command line only. Anaconda provides a lot of software, also for R, you may explore it by starting the graphical user interface. (we do not dig deeper here).
- Miniconda is an alternative to Anaconda. It is smaller and just comes with the core libraries which are necessary to use conda.

Anaconda installation

- Run setup program
- When asked check the checkbox to set the PATH variable
- Open git bash and type

  `conda init bash`
- If this works, you successfully installed conda!

Using conda environments

- A conda environment allows you to install a separate version of Python (or any other conda supported software) with associated packages
- List all available environments

  `conda env list`
- There should be a *base* environment available on all installations. It is automatically activated when you use conda!
- We now first use conda to *update* - conda!

  `conda update conda`
- This will download and install the newest version of conda

Creating and activating environments

- You can create a new Python environment with this command

  `conda create -n <environment-name> python=3.7 anaconda`
- *<environment-name>* can be chosen by you. We use *scientific-computing*:

  `conda create -n scientific-computing python=3.7 anaconda`

- Again list all available environments

  `conda env list`
- There should be an environment *scientific-computing* available now.
- How to work with it?

  `conda activate scientific-computing`
- Observe how the command line indicates the active environment: *(scientific-computing)* is now displayed
- Not activating the correct environment is a very common source of problems! Check your environment!

Jupyter

What is a Juypter Notebook?

- In principle, you can write your code in any test editor and then run it on the command line
- A much better option would be to use an integrated development environment (IDE) such as PyCharm (syntax highlighting, code linting, etc.).
- A different form of coding environment are so called *Jupyter Notebooks*
- Here, the code is written in the web browser, sent to a server, executed there, and results are displayed in the notebook again.

Pros & Cons of Jupyter Notebooks

- Advantages
  - Integration of code and visualization of results: for data analysis crucial.
  - Interactive environment for data assessment: data (variables, objects, libraries) stay in memory and code can be dynamically adapted (Similar to R)
  - Code an be run on server with large computational capacity (if available)
- Disadvantages
  - Managing large junks of code (such as functions) becomes complicated.
  - Using Jupyter notebooks with version control is difficult, as they introduce their own XML-syntax, which is not really human readable.
  - Interactive session: very error-prone, as objects and variables can change their state depending e.g. on the number of times a certain cell is executed.
- Best of two worlds: use an IDE to manage your stable code (functions etc.) and use notebooks for exploration.
- For teaching purposes, notebooks are VERY useful. We are going to use them therefore - and may, at some point, introduce additional software such as an IDE.

Using Jupyter Notebooks

In git bash, type

```
jupyter notebook
```

This will start the Jupyter Notebook server within the current working directory as a background process and starts a browser.
To stop it type

```
CTRL-C
```

If you want to use conda environments in your notebooks, do the following:

- Install *ipykernel* in your environment

  ```
  conda activate <conda-environment>
  conda install ipykernel
  conda deactivate
  ```

- Additionally, install *nb_conda_kernels* in *base* if you run your jupyter notebook from base:

  ```
  conda activate base    # could be also some other environment
  conda install nb_conda_kernels
  ```

Standard Browser and Jupyter Notebooks

jupyter notebook may start with an unwanted Browser. Two things you should try:

- Set your desired Browser as Standard Browser in Windows Settings (Control Panel - Apps - Default Browser)
- try

  ```
  git config --global web.browser google-chrome
  ```

Some interesting links

- Jupyter notebook cheatsheet
- Free Python class
- The Hitchhiker's Guide to Python

Homework assignment

Homework assignment (I)

- During your last homework assignment, one of your team members forked the homework repository: https://github.com/inwe-boku/homework-scientific-computing
- Fetch the latest changes from the upstream:

```
cd path/to/homework-scientific-computing
git remote add upstream https://github.com/inwe-boku/homework-scientific-computing/
git pull --no-edit upstream master
```

- Install conda and Jupyter on your computer, run Jupyter in the directory of the forked repository.
- Create a new Notebook and use Python to create an encrypted file with your full name, the registration number (Matrikelnummer), and the git username. For that purpose, use the public key *public_key.pem* stored in the homework repository in *homework02-conda-python*
- The filename of the encrypted file should be *group-member-<github-account-name>*, where *<github-account-name>* is your Github name.
- Use the Python code on the next slides to accomplish this task.
- Add the encrypted file to your repository, commit the change, and push it to your fork (no pull request).

Homework assignment (II)

```
"""Store personal data encrypted with our public key."""

import subprocess

# if you get an ImportError, you missed to install
# the package cryptography with conda
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import padding

# change this:
real_name = "First Last"
github_account = "mynickname"
regstration_number = "123456"
```

Homework assignment (III)

```
# open and read the public key
with open("public_key.pem", "rb") as key_file:
    public_key = serialization.load_pem_public_key(
        key_file.read(), backend=default_backend())

def git_config_value(param):
    raw_value = subprocess.check_output(
        ['git', 'config', f'user.{param}'])
    return raw_value.decode().strip()

# determine git name/mail address
git_name = git_config_value('name')
git_mail = git_config_value('email')

# concatenate strings as CSV format
name_mapping = ";".join((real_name, github_account,
                         regstration_number, git_name, git_mail))
```

Homework assignment (IV)

```
# encrypt string
encrypted = public_key.encrypt(
    name_mapping.encode(),
    padding.OAEP(
    mgf=padding.MGF1(algorithm=hashes.SHA256()),
    algorithm=hashes.SHA256(),
    label=None))

# write encrypted data to disk
out_filename = f"group-member-{github_account}.txt"
with open(out_filename, 'wb') as f:
    f.write(encrypted)
```